Interactive Physics Version 1.2

A. John Mallinckrodt

Citation: Computers in Physics **5**, 349 (1991); doi: 10.1063/1.4822995
View online: https://doi.org/10.1063/1.4822995
View Table of Contents: https://aip.scitation.org/toc/cip/5/3
Published by the American Institute of Physics

PRODUCT REVIEW

Interactive Physics Version 1.2

Interactive Physics Reviewed by A. John Mallinckrodt

Interactive Physics Version 1.2, system requirements: Macintosh Plus or higher, a hard disk or 2 disk drives, and System 6.0.2 or higher. Available from: Knowledge Revolution; 497 Vermont Street, San Francisco, CA 94107, (415) 553-8153, KR@AppleLink.Apple.com. List Price: $249. Lab Packs available.

Interactive Physics (*IP*) provides the user with a computer-based laboratory that simulates the mechanical behavior of objects in a two-dimensional world. Its most seductive feature is that the primary method of data input and output is graphical. For the most part, the user literally draws the experiment and its initial conditions. *IP* then calculates and displays the consequent action in the form of an animation.

Providing such capability is a tall order indeed—at least if one cares to get the physics right—and *IP* is not entirely successful. Nevertheless, its usefulness as a demonstration aid and as an instrument for a myriad of student assignments and mini-research projects make it well worth the cost. *IP* will teach *anyone* some physics.

In this review, I intend to present a reasonably detailed examination of how *IP* works in order to expose both its strengths and weaknesses. To prepare readers who are unfamiliar with *IP* for what follows, the remainder of this section is devoted to an overview of the program's operation. Readers with previous experience of using *IP* may prefer to skip to the next section.

In preparation for drawing an experiment, the user adjusts properties of the "world" in which it is to take place. The world's size may be adjusted from as large as one square kilometer down to one-hundredth of a square meter and scroll bars and zoom commands can be used to view all or any part of the world. A "downward" (i.e. toward the bottom of the screen) gravitational field vector can be adjusted in magnitude from 0 to 25 N/kg, and a very rudimentary form of quadratic air resistance can be invoked by setting what amounts to a universal drag coefficient for which no units are specified. (See the discussion of air resistance in the section entitled "Could It Be Any Better?".) *IP* provides toggles for the display of coordinate axes, a coordinate grid, and horizontal and vertical rulers.

The user creates an experiment by drawing "mass objects" in the form of squares, rectangles, circles and polygons, using standard Macintosh drawing tools. A "grid snap" option is provided, allowing for more control over the positions and sizes of objects by forcing their vertices (horizontal and vertical extrema in the case of circles) to occupy points on a coarse lattice. Once they're drawn, the objects can be resized, repositioned, rotated and colored. Each object carries a user-specifiable name, and is characterized by an adjustable mass and two adjustable—but unquantified—parameters, which *IP* calls the "Surface Parameter" and the "Elasticity."

Whenever two objects come into contact, these parameters work in combination to determine, in an empirical manner, the frictional and rebound characteristics of the interaction. Alternatively, one may select specific pairs of objects and explicitly specify their mutual coefficients of friction. The initial center-of-mass position, rotational orientation, and translational and rotational velocities of each mass object can be set in a spec sheet accessed by double-clicking on the object. When precision is less important, the initial translational velocity of an object may be specified graphically by selecting the object and dragging (from its center-of-mass) a vector of the desired size and direction. Any object may be anchored in space to form a fixed platform or barrier.

IP can display vectors representing an object's velocity or acceleration; the individual forces exerted on it by other objects, air resistance or gravity; or the *net* force acting on it. In addition, the user can request meters which monitor these quantities as well as time, position, rotational position, velocity or acceleration, translational or rotational kinetic energy, translational or rotational momentum, net torque, and the forces of contact between any two selected objects. The meters can provide digital or analog readouts or strip-chart-type recordings of the monitored value.

The user may use drawing tools to connect mass objects to each other and/or to fixed points in space with "no-mass objects"—strings, springs, and dashpots. These objects have appropriate adjustable parameters, which, again, can be accessed by double-clicking or via a menu item. They apply their forces at the points to which their ends are connected. No-mass objects do not interact with objects other than those to which they are connected, so that, for instance, one cannot form a pulley by wrapping a string around a circle.

A final drawing tool produces arbitrary constant force vectors which can be applied at any point on mass objects and which can maintain either their absolute orientation in space, or their orientation with respect to the object itself.

When finished with the set-up of an experiment, the user clicks on "Run", and *IP* animates all objects in accordance with Newton's laws, using a simple fixed time-step numerical integration method. (See the next section for a brief discussion of the computational algorithm.) The user can specify the integration time-step. As with all numerical integration schemes, the simulation accuracy (generally) increases

A. John Mallinckrodt is an Associate Professor of Physics at Cal State Polytechnic University, Pomona, in Pomona, CA 91768.

with smaller time-steps at the necessary expense of simulation speed.

Because substantial amounts of time are spent doing the physics, *IP* records animation frames in a "tape player memory" which can be used to replay the animation forward or backward, either continuously (at increased speed) or one step at a time. Tape player memory can be allocated in chunks from 25 kbytes up to the limit imposed by the amount of memory available to the program. The default allocation of 400 kbytes is sufficient to store about 4000 frames of a simple, two-object experiment, but will manage only about 120 frames of the most complex experiments. An optional warning dialog box pops up when the memory is first filled and, if the user allows *IP* to proceed, new frames will overwrite the earliest frames, preserving only the most recent segment of the experiment. A "Skip Frames" option allows the user to view only one frame out of every two, four, eight or 16 to further speed up replayed experiments. Finally, a "Tracking" option leaves unerased every first, second, fourth, eighth, sixteenth or thirty-second frame, giving a stroboscopic picture of the experiment. Any frame (or tracked sequence of frames) can be printed.

At the end of an experiment, the user may "export" the data collected by all meters, to a data document which can be analyzed later using a spreadsheet or some other type of data analysis program. Alternatively, data from any set of the available meters may be copied to the clipboard and immediately "pasted" into the analysis program.

IP comes with two folders of example scenarios. The "Demonstration" folder contains a set of simple experiments providing a quick tour of *IP*'s capabilities. The "Experiments" folder offers experiments in one- and two-dimensional kinematics, dynamics, energy, momentum, collisions, rotational dynamics, equilibrium, and oscillations which are intended as the basis for a variety of student "lab" exercises. These scenarios stick to relatively simple situations and are probably most useful as jumping-off points for student investigations or demonstrations. A supplementary book entitled *Physics Interactions* serves as a lab manual for a set of experiments on an accompanying disk.

How Does It Do It?

As is true for any numerical simulation, the user must be alert for unphysical artifacts in the output of *IP* that result from the fundamental limitations of its computational algorithms. Without some understanding of what those algorithms are, one can't make fully intelligent use of *IP* and will inevitably be frustrated by a number of ill-mannered and unphysical behaviors.

For its routine operations, *IP* uses what amounts to a modified, fixed time-step, Eulerian integration scheme. At the beginning of each integration step, *IP* determines the individual forces and their points of application, and uses this information to find the net force and net torque acting on each object. Dividing these quantities by the mass and rotational inertia gives the translational and rotational accelerations. The translational and rotational impulses are calculated—under the assumption that the forces and torques are constant over the duration of the next integration time-step—and added to the old translational and rotational momenta to obtain new values. Dividing these momenta by the mass and rotational inertia of the object provides the new translational and rotational velocities.

Finally, *IP* calculates the new translational and rotational positions, assuming that the *new* values for the velocities were in effect throughout the previous time-step. This modification of the standard Euler method has been shown to yield more accurate results for a given step size in many cases of interest.[1]

The foregoing procedure would be simply executed if it were not for complications that arise primarily in determining the appropriate magnitudes and effective points of application for forces of contact. These problems only *begin* with trying to determine when and where pairs of objects come into contact in the first place. They become especially acute when one cares to model the effects of friction and partial elasticity with any semblance of physical correctness. However, in many circumstances, the last straw is the commitment to do all of this with a constant integration step size for the sake of both maximizing and maintaining animation speed. Forces of contact are particularly unforgiving in this regard.

IP adopts several empirical rules which, in effect, assert desired outcomes to determine requisite forces. For instance, at least in the simple case of a collision between two objects, the product of the elasticity parameters determines the ratio of the relative velocity *after* the collision to that *before* the collision. During the first integration step after *IP* detects a collision—which will, in general, be *after* the drawn boundaries of the two objects visibly overlap—it applies equal and opposite normal forces calculated to provide the impulse necessary to yield the desired final velocities in a single time-step.

Additional complications arise from *IP*'s need to remember that it has "taken care of" the collision even though the boundaries will, in general, continue to overlap for one or more frames. Of course, it can simply determine whether the relative motion of the objects is toward or away from each other, but what happens if other objects are simultaneously interacting—pushing each other together? As you may see, it can get complicated.

Another empirical rule is used to treat initially-stretched ropes. When a string is found to be longer than its resting length, *IP* applies a single-step impulse to whatever objects may be found connected to its ends. The impulse is calculated on the basis of a desire to return the string to its resting length in a relatively small number of frames and, therefore, depends not only on how much the string is stretched, but also on the masses and velocities of the objects to which it is attached, and the integration time step. Furthermore, if *IP* determines that the string is making sufficiently rapid progress toward its resting length, the string will apply no further forces to the objects, regardless of the fact that it may still be substantially stretched. In situations where

other forces work against the desires of the string, *IP* will reach a compromise in which the string exerts whatever force is necessary to effect a damped return to its resting length. Although one may find this algorithm disturbing on physical grounds, in most cases it is probably preferable to a more realistic one in which the string would merely break if it were stretched by any noticeable amount.

Any Problems?

In the course of putting *IP* through its paces, I discovered a number of bugs. In a conversation with David Baszucki, president of Knowledge Revolution and author of *Interactive Physics*, I was informed that the most important four of these have been exterminated and will not be found in Version 1.2.1, which is now shipping. He also told me that registered owners of Version 1.2 will receive the update free of charge. The four exterminated bugs were characterized by the following pernicious behaviors:

- Causing the effective coefficient of friction between two objects to behave as though it were four times what the user had specified.
- Losing track of the area of objects which have been resized. This error also affected the calculation of the object's rotational inertia.
- Including two spaces before each numerical datum (only one space in front of negative values) in the tab-delimited data export files. Although not really a bug, these spaces caused some spreadsheets and data analysis programs to consider the data as text, and necessitated some form of conversion before numerical manipulations could be accomplished.
- Failures of the contact detection algorithm that occasionally allowed objects under certain seemingly benign circumstances to pass directly through each other.

Other bugs I encountered—many of which the people at Knowledge Revolution were apparently previously unaware of—included the following:

- In at least one rather special situation, the frictional interaction between two objects acts in a direction which would *increase* the relative velocity between the points of contact.
- Another failure of the contact detection algorithm that sometimes occurs when a collision between two objects takes place at vertices of both objects.
- Displaying reversed normal and friction force vectors in certain circumstances. This appears to be the result of the program becoming confused about which of a given contacting pair has been selected for vector display. Saving such an experiment and reopening it causes the vectors to get assigned properly.
- A tendency for the air force vector to lock up when an experiment is run backwards, and to disappear if an experiment is reset and run from the start.
- A lack of communication between the surface parameter that is displayed for individual objects, and the coefficient of friction that is displayed when two objects are simultaneously selected. When one sets the surface parameters and subsequently examines the coefficient of friction between the two objects, it will be displayed as zero even though running the experiment reveals that it certainly is not.
- Reminiscing strip charts. On the initial run of an experiment with a strip-chart meter, the strip-chart will often lose track of the current value being monitored and begin replaying data from the beginning of the experiment. The problem usually disappears if the experiment is reset and run again.
- WYSIWYG failure. Printed versions of experiments are not always identical to screen versions. In one case, a vector labeled (correctly) "FN" on the screen was printed as "FG."

Note that most of these are relatively benign bugs that do not affect the simulation itself. Anyone working with *IP* for a reasonable amount of time will stumble onto other odd behaviors. Most of the time these are not so much bugs as they are simple artifacts of the numerical methods used by *IP* to model interactions. Many of these artifacts can be tamed simply by reducing the integration step-size. For instance, if the single-step impulse on an object due to air resistance is larger than its current momentum, then air resistance acting alone will instantly *reverse* the object's direction of motion. Reducing the integration step size will reduce the very large error caused by *IP*'s assumption that the drag force is constant over that interval.

A more subtle artifact sometimes emerges if the integration step is too small. Evidently in the interest of calculational speed, *IP* truncates most kinematic data at a predetermined decimal place intended to limit the number of significant digits that it has to carry around. As a result, if an impulse is non-zero, but too small to cause a one-digit change in the related momentum in a single step, then the momentum will not be altered by the impulse. Short of changing the forces on, or inertia of, the object, only increasing the integration time step will eliminate this pathological behavior. Truncation error also pops up in situations where small consecutive changes in a parameter may not affect the experiment at all until the parameter is changed by a critical amount. At that point, a substantial alteration of the experiment may be observed.

The moral of these stories is that one must be aware of the algorithms used and be alert for their effects—a fact of life in all computational work.

Any Tips?

Experiments involving objects which slide along each other's surfaces are complicated by initial bounces that result from not being able to position or rotate the objects so that their surfaces are precisely aligned and just touching. There is, however, a nice way to obtain such initial conditions: turn the elasticity parameters of both objects way down and let the experiment run just long enough for the bouncing to cease. Stop the experiment and choose "Start Here" from the "Experiment" menu. The current (bounceless) conditions become the new initial conditions. Now you can re-adjust the elasticities and other parameters.

The color option may be appealing to users having color monitors, but there are few situations in which it is genuinely helpful, and it should be understood that the computational

overhead involved in keeping track of the extra color bits exacts its price in animation speed. In regular use, the monitor cdev (in the Macintosh Control Panel) should be used to select the

Color Mode	1 bit	2 bit	4 bit	8 bit
Simple task	2700	2000	1100	400
Complex task	520	450	380	220

Table 1. Animation speeds (in frames per minute) for two *IP* tasks running in 1-, 2-, 4- and 8-bit monitor modes on a Mac IIci. The "simple task" was a ball bouncing on an anchored platform. The "complex task" involved nine objects inside an anchored container object connected with three springs and a dashpot.

1-bit (or "Black & White") monitor mode. When color *is* required, limit yourself to the 2-bit (or "four-color") mode. Table 1 provides a comparison of approximate animation speeds on my Mac IIci (in frames per minute) for two tasks—one simple and the other complex—running in the 1-bit, 2-bit, 4-bit and 8-bit monitor modes. Note that there is a much larger relative penalty for simple tasks than for complex tasks—a result of the more nearly-fixed time (regardless of animation complexity) required to update the display after each integration step.

Could It Be Any Better?

No question. From the moment *IP* was released, Knowledge Revolution has been on the receiving end of a host of "Why-doesn't-it's", "It-needs-to's", and "I-have-to-have's." A sampler of items from my own list follows.

First, a number of modest requests (in rough order of the importance I would attach to them):

- Oscillating and/or revolving drive points to which one can attach strings, springs and dashpots, and the ability to "track" an experiment at constant drive phase. (This is absolutely the number one request in my book. This simple addition would enable a world of fascinating experiments in nonlinear dynamics.)
- A supplemental reference guide which is *substantially* more forthcoming about the details of the numerical algorithms. As I have insisted, information of this kind is essential for the intelligent use of *IP*.
- Pivot points to provide torqueless rotation of an object about any point. (This is a necessary first step toward providing pulley capabilities—a more difficult proposition.)
- Numerically specifiable surface and elasticity parameters. (These omissions—the result of a conscious decision to shield users from the quantitative aspects of models deemed too unphysical to merit the costume of numerical precision—only diminish one's ability to use the program intelligently and reproducibly.)
- A display of rotational inertia in the spec sheet of mass objects, and the option of adjusting it up to the physical limits imposed by the mass and physical size of the object. (At present, the rotational inertia about its center-of-mass is calculated correctly—even for arbitrarily shaped polygons—under the assumption that the object's mass is uniformly distributed over its area; however, it is not displayed or separately adjustable.)
- Velocities specifiable by magnitude and direction. (For instance, in projectile motion experiments, one would really like to change the initial angle of projection without altering the speed.)
- Meters which are able to display vector quantities in magnitude/direction format.
- Keyboard equivalents for virtually all menu items, as well as for selecting the drawing tools.
- A "Skip Frames" option that is active during the initial run. (At present, animation frames can be skipped only during replay. The speed and accuracy of some simulations would benefit dramatically if one could reduce the integration time step and request that *IP* refrain from drawing most frames.)
- All sorts of meters beyond the admittedly substantial number already provided. In particular, I would like a "Total Energy" meter which could be used to monitor experiments devoid of friction and inelasticity as a crude check for the build-up of numerical errors.
- Optional numerical control over the positions of the no-mass objects. The user should be able to specify explicit positions for the "business ends" of all no-mass objects. (In the case of strings, springs, and dashpots, one *can* at least force them to be connected to objects at their centers-of-mass; no such facility is provided for force objects.)
- Reshapeable polygons.

Next, a wish list of items which would most likely take longer to be implemented:

- Pulleys. (This is not as simple as it may sound since it would require strings to be imbued with the ability to detect contact with mass objects along their entire periphery. As soon as we begin thinking along these lines, we will want *non*massless strings which support waves. Lots of calculations going on now!)
- Sticky surfaces that can support an attractive normal force, perhaps up to some maximum value. (This is also likely to be harder than it sounds, due to the complications it will introduce into the friction and elasticity algorithms.)
- A much better air resistance algorithm. Air resistance is an interaction that is exceedingly difficult to model correctly. To do so *and* maintain any real-time animation capability is nearly impossible. *IP* makes some serious compromises here and implements air resistance using an algorithm in which the force is given by $\mathbf{F}_{air} = -kv^2 l_\perp \hat{v}$ where k is the air resistance parameter set by the user (and applicable to *all* objects in any given simulation), v is the translational velocity of the body's center-of-mass, and $l_\perp$ is the length of the body's projected section perpendicular to its direction of motion. The force is directed opposite the velocity and is applied at the center-of-mass.

As a result, semi-realistic results are possible only in the case of non-

rotating bodies which possess and maintain symmetrical aspects in the direction of motion. For instance, a rotating object will experience no decelerating torque and a moving asymmetrical object will neither be deflected nor rotated by the effects of air resistance. Don't even *think* of trying to simulate a knuckle-ball!

On the other hand, the algorithm is adequate enough for simulating the motion of a simple projectile or the deceleration of a disk moving along a frictionless surface. Even accepting its current limitations, the manual should disclose the algorithm being used, its severe limitations, and the units of k (which are kg/m^2).

- A way in which to make the gravitational field vector point into the screen with objects sliding and rotating frictionally on a two-dimensional surface. This is a *highly* non-trivial problem. The frictional forces and torques exerted by a surface on a rotating and translating body are very complicated as a result of the fact that each element of the contact surface is moving in a different direction. Of course, this is why we want a computer simulation to do it for us!
- The ability to give objects arbitrary charges—or, better yet, charge distributions—and to specify external electric and magnetic fields. As long as we're at it, let's have a magnetic dipole moment as well.
- A gravitational interaction between objects, perhaps with variable dependence on distance, for exploring orbital mechanics. (Here I must briefly mention *Gravitation, LTD.*, a remarkable shareware program written by Jeff Romereide, 343 Elma Avenue, Laurel Springs, NJ 08021. Although it deals with a more limited realm of phenomena than *IP*, *Gravitation, LTD.* is another must-have piece of software.)

The Knowledge Revolution people are well aware of all these desires and, not surprisingly, have dreams of their own that go well beyond. Nevertheless, it is important to keep in mind that substantial new capabilities will almost certainly come at the cost of both simple operation and animation speed.

What Is It Good For?

First and foremost, *IP* deserves a place in every mechanic's classroom as a device that bridges the gap between the blackboard and traditional demonstrations. The blackboard is a static medium, woefully inadequate when it comes to conveying dynamic information. On the other hand, demonstrations are notoriously treacherous, and even on a good day it's not possible to vary all the parameters one might like. With a projection system and a moderate investment of time, an instructor can use *IP* to great advantage as an in-class supplement.

IP can also become the research instrument for an unlimited number of student-guided mini-research projects. For instance, Fig. 1 shows a frame from a quickly designed and run experiment to investigate the statistical behaviors leading to the equipartition of energy in an ideal gas. A "gas" consisting of eight 1-kg disks and one 4-kg disk is confined inside a container. The initial velocities of all objects except the 4-kg disk are zero. The 4-kg disk is given an initial velocity of 5 m/s (KE = 50 J) toward the others. All collisions are frictionless and perfectly elastic. Running the experiment provides a dramatic, visually-delivered lesson on how a gas is "thermalized." It argues poignantly in favor of the second law of thermodynamics, and it makes clear that the average speed of the large disk and the relative *variations* of its speed are smaller than those of the others.

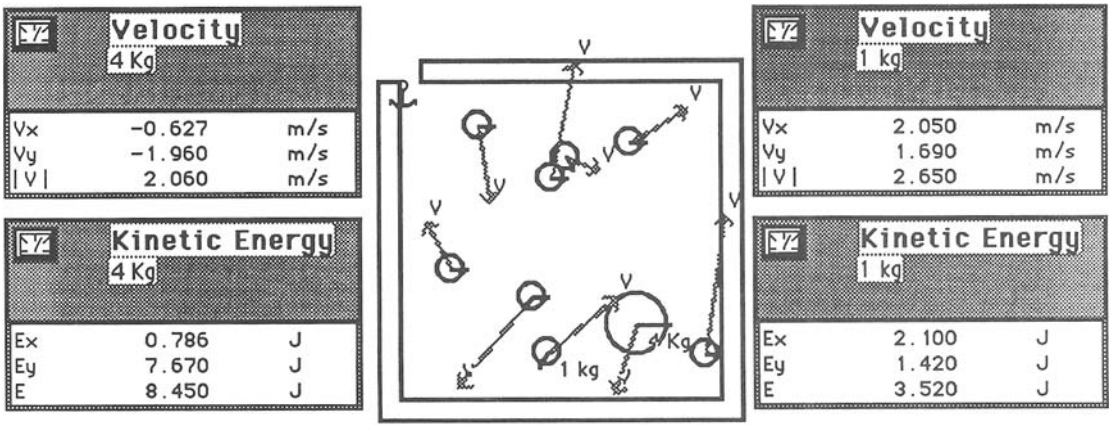

Fig. 1. A frame from an experiment designed to explore equipartition of energy. Eight 1-kg disks are initially at rest, but confined within a gravity-free box with a single 4-kg disk which has an initial speed of 5 m/s. All interactions are perfectly elastic and frictionless. (The total kinetic energy of the system is 50 J or about 5.6 J per object.)

Furthermore, by metering the large disk and one of the small disks, and exporting the data for external analysis, one can perform calculations which augment the observations. Figs. 2 and 3 show, respectively, the speeds and kinetic energies of the two disks as functions of time and the averages and standard deviations of these quantities. Note that the average speed of the small disk is twice that of the large one, while the energies are more comparable. The average energy of the large disk, which experiences smaller variations, is very

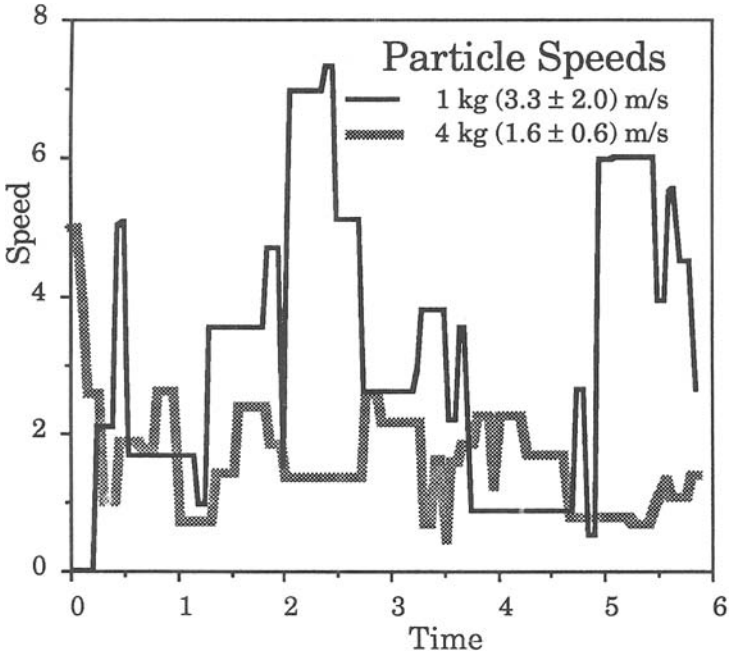

Fig. 2. Data from the experiment of Fig. 1. The speeds of one of the 1-kg objects and the 4-kg object are plotted as functions of time. The legend gives the average and the standard deviation of the values for each object. The first few data points have been left out of the statistical calculations to allow the system to "thermalize." This figure and Fig. 3 were created by exporting data from *IP* into Cricket Graph.™

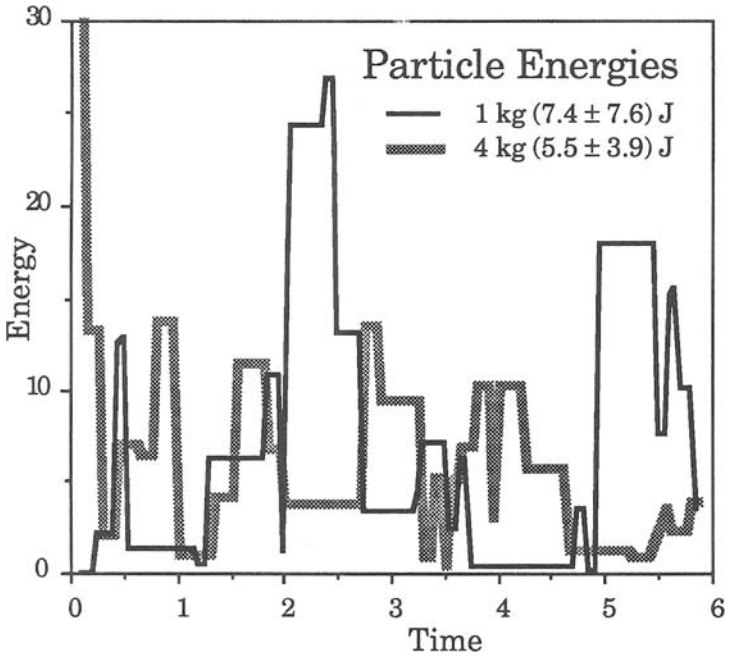

Fig. 3. Data from the experiment of Fig. 1. The energies of one of the 1-kg objects and the 4-kg object are plotted as functions of time. (See the caption for Fig. 2.)

close to one-ninth the total kinetic energy of the gas (50 J/9 = 5.6 J). A number of other statistical predictions can be checked by looking at the absolute and relative variations in the speeds and energies.

Computer-based research projects like this combine the essential elements of experimental and theoretical research. They provide time-efficient methods for students to explore phenomena and to collect data unencumbered (for the moment) by enormous instrumental roadblocks. The data can be analyzed empirically for trends and/or examined for conformity with theory. Although the example of the last paragraph was intended to show off some of the abilities of *IP*, and might be better suited to students who have completed their introductory physics courses, similar but less ambitious projects are appropriate even for beginning students and can foster an early appreciation of the excitement of independent research.

What Do You Think Overall?

All things considered, I like *IP*. Although my wish list is long, my admiration for the capabilities already there is substantial. Yes, it has bugs and, yes, even *beyond* the bugs it can easily get in over its head. One has to be aware of its limitations and know how to work around them. On the other hand, more than once have I learned some physics by spending a little time thinking about an odd result that I had believed at first to be the result of a flaw in the program. And even when it *is* wrong, valuable lessons can be learned from considering the cause of the failure; there are remarkable subtleties involved in applying Newton's laws to the interactions of extended bodies. The more I use *IP*, the more things I think of to do with it. It supports a wide range of applications, and I can imagine no readier form of simulation set-up. Get your hands on it and come to your own conclusions, but if you are an instructor of high school or lower-division college physics—or if you simply enjoy a headier sort of computer game—you will want to learn about this program. ■

References

1. A. Cromer, *Am. J. Phys.*, Vol. 49, 1981, p. 455.